Interpolation-based Classifier Generation in XCSF

Anthony Stein, Christian Eymüller, Dominik Rauh, Sven Tomforde and Jörg Hähner
Organic Computing Group, University of Augsburg, Germany
{*anthony.stein* | *sven.tomforde* | *joerg.haehner*}@informatik.uni-augsburg.de

***Abstract*—XCSF is a rule-based on-line learning system that makes use of local learning concepts in conjunction with gradient-based approximation techniques. It is mainly used to learn functions, or rather regression problems, by means of dividing the problem space into smaller subspaces and approximate the function values linearly therein. In this paper, we show how local interpolation can be incorporated to improve the approximation speed and thus to decrease the system error. We describe how a novel interpolation component integrates into the algorithmic structure of XCSF and thereby augments the well-established separation into the performance, discovery and reinforcement component. To underpin the validity of our approach, we present and discuss results from experiments on three test functions of different complexity, i.e. we show that by means of the proposed strategies for integrating the locally interpolated values, the overall performance of XCSF can be improved.**

I. Introduction

Learning Classifier Systems (LCS) have gained plenty of research interest in the past four decades since their introduction by Holland in [1]. He was the first who combined the powerful *genetic algorithms* (GA) with a rule-based learning system that iteratively strengthens its evolved rules. Wilson enhanced Holland's initial approach by simplifying certain aspects in 1994 [2]. With his *Zeroth-level Classifier System* (ZCS), the broader family of Michigan-style LCS' had become a new relative that made the fundamental aspects tangible for succeeding research. Only one year later, Wilson in [3] extended his ZCS towards the today very broadly investigated and well-applied *eXtended Learning Classifier System* (XCS). His fundamental change in thinking about the issue which classifier suits a situation most accurately, constituted a major step towards the research that succeeded until today. Wilson suggested to distinguish between the *strength* of a rule and its *accuracy*. Within a certain environmental niche, more accurate classifiers compete for reproduction in the GA, and thus exert evolutionary pressure towards a population of accurate rules instead of rules that only deliver high payoff. Due to the additional reproduction within an environmental niche and deletion from the entire population of evolved classifiers, XCS attempts to construct not only accurate but also maximally general rules. This aspect became known as Wilson's *Generalization Hypothesis* [3]. Kovacs extended Wilson's hypothesis in [4] to the so-called *Optimality Hypothesis*, claiming that XCS is reliably capable of learning optimal solutions for multiplexer problems. Initially only for binary problem spaces, later on XCS was extended to cope with real-valued inputs, i.e. $x \in X \subseteq \mathbb{R}^n$ in [5]. This further extension became popular as XCSR and paved the road for an even broader domain of learning problems, such as regression or classification in real-valued domains. A lot of analytical research focused on the capabilities of XCS and found that XCS reliably evolves accurate, complete, compact and maximally general mappings of inputs $x \in X$ and according actions $a \in A$ to predicted outputs $p \in P$ ($X \times A \Rightarrow P$) [3]. But even if XCS generally achieves the construction of a completely covered problem space, there are at least two challenges remaining: (1) Not all niches of the problem space can be expected to be sampled equally frequently, which in turn delays the construction of a complete map. (2) Uncovered environmental niches are reactively handled by a so-called *covering* mechanism, which fills the gap more or less sufficiently and initializes a classifier with predefined, possibly arbitrarily chosen initial values.

This paper provides a general idea of how XCS' well-understood algorithmic structure can be extended by means of *local interpolation*. With the proposed interpolation-based approach, we incorporate knowledge in terms of already existing classifiers. We strive to weaken the negative effects of uncovered environmental niches by taking neighboring classifiers into account for a more appropriate classifier creation. Consequently, the proposed techniques affect the overall evolution of the entire population. Therefore, we introduce two strategies for integrating interpolated knowledge into fundamental algorithmic steps along XCS' main-loop. On the one hand, we extend the covering mechanism to prevent possibly arbitrarily chosen initialization values. On the other hand, the genetic algorithm is slightly modified. Instead of solely relying on the parents' classifier attributes, we also take the already evolved and reinforced values of each classifier that belongs to the same environmental niche into consideration.

The remainder of this paper is structured as follows: A brief description of the fundamental functioning of Wilson's initially presented XCS for function approximation (XCSF) is outlined in Section II. Section III introduces a local interpolation technique called *Modified Shepard's Interpolation*. Additionally, this section discusses how interpolation can be integrated into XCS' well-known algorithmic structure and presents two novel integration strategies. Section IV reports results of conducted experiments and demonstrates the general potential of using interpolation techniques in the algorithmic structure of XCSF. Eventually, Section V concludes the paper with a short discussion and some aspects of our current efforts.

II. Standard XCSF in Brief

The following paragraphs sketch the general architecture and algorithmic steps of Wilson's initially introduced XCS for

function approximation (XCSF) [6]. In the scope of this paper, we limit our elaborations to this initial variant as it is sufficient to provide a general idea of how interpolation techniques can be integrated into XCS' algorithmic structure. Consequently, we yet incorporate the axis-parallel hyper-rectangular condition representation as described further by Butz in [7]. A variety of relevant extensions, briefly described in Section II-B, are under current implementation. The combination of the presented approaches with the major extensions of standard XCSF constitutes a top priority of our research agenda.

A. Algorithmic and Architectural Structure of XCSF

XCSF is a rule-based learning system that evolves so-called classifiers at runtime and iteratively updates their attributes. In contrast to conventional XCS variants, XCSF strives to approximate functions in a piecewise linear fashion. The knowledge-base of XCSF is called population $[P]$ of classifiers cl whose size is restricted by a maximum number N. A single classifier cl comprises a couple of attributes: (1) A condition C that defines a particular subspace of the problem space, defined by $C = (\vec{l}, \vec{u}) = (l_1, \ldots, l_n)^T, (u_1, \ldots, u_n)^T$, where l_i and u_i specify the lower and upper bounds, respectively, for each of the n dimensions. (2) Usually, an action $a \in A$ defines a reaction to be executed on the environment. In the case of XCSF, only one so-called *dummy action* a_d is defined to adhere to the conventional XCS structure. (3) A payoff prediction scalar p that is determined by a linear function $h(\vec{x}_t) = (\vec{x}_t^* - \vec{l}^*)^T \vec{w}$, each time a new situation vector, or rather sample $\vec{x}_t = (x_1, \ldots, x_n)^T$ arrives the system. $\vec{w}$ is defined as an $n+1$ dimensional weight vector $(w_0, \ldots, w_n)$, where w_0 acts as offset weight and $\vec{x}_t^*$ and $\vec{l}^*$ represent $\vec{x}_t$ and $\vec{l}$ but with a leading 1 and 0, respectively. (4) The ϵ attribute determines the mean absolute prediction error of that particular classifier. (5) The relative accuracy of a classifier is denoted as fitness F, which can be interpreted as sort of inverse function of ϵ. More formally, in the case of XCSF the relative accuracy κ' calculates as follows:

$$\kappa' = \frac{cl.\kappa \cdot cl.num}{\sum_{cl_j \in [M]} cl_j.\kappa}, \text{ where } cl.\kappa = \alpha(cl.\epsilon/\epsilon_0)^{-\nu} \quad (1)$$

$cl.num$ defines a classifier's numerosity, i.e. how often the classifier has subsumed a similar classifier (cf. [3]). Thus, F defines the accuracy of the classifier's prediction relative to competing classifiers within the same environmental niche, i.e. the set $[M]$ of classifiers matching the current situation. Each classifier also maintains an experience attribute $cl.exp$ that indicates how often a particular classifier has been triggered for execution. For a more detailed description of additional parameters, the reader is referred to e.g. [3].

Figure 1 illustrates the three main components (*performance*, *discovery* and *reinforcement*) of XCSF and further extends the conventional structure by the novel interpolation component (IC). Additionally, a single iteration through XCSF's main-loop is sketched.

At each timestep t a situation vector $\vec{x}_t$ is retrieved from a function $f : X \to \mathbb{R}, \vec{x}_t \to y, X \subseteq \mathbb{R}^n$. A *matching* procedure

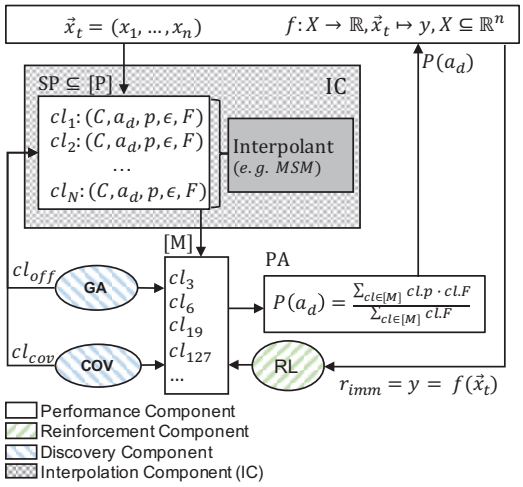


Fig. 1: Schematic illustration of XCSF augmented by the novel *Classifier-based Interpolation Component* (CIC)

scans $[P]$ for all matching classifiers and forms a so-called *match set* $[M]$. A classifier cl_i matches a current input $\vec{x}_t$, iff $l_i \leq x_i < u_i$. After $[M]$ is formed, each $cl \in [M]$ recalculates its prediction $cl.p$. Subsequently, the *system prediction* $P(a_d)$ is calculated by the *prediction array* PA as follows:

$$P(a_d) = \frac{\sum_{cl \in [M]} cl.p \cdot cl.F}{\sum_{cl \in [M]} cl.F} \quad (2)$$

The next step constitutes a major difference between XCSF and conventional XCS variants for classification. Instead of an action a that is actualized on the environment the learning classifier system acts on, the system prediction $P(a_d)$ is used as output. This is also the reason for the missing *action set* $[A]$, since for XCSF holds $[M] = [A]$. Further, no designated payoff function that delivers an immediate reward r_{imm} is needed, but rather the actual function value $f(\vec{x}_t)$ is delivered in return. Thus, we assume that at the latest in the succeeding time step $t+1$ the actual function value of the situation vector of time t is going to become known. With the actual $f(\vec{x}_t)$ at hand, XCSF calculates the absolute error of the predicted payoff $P(a_d)$. This absolute error is then used to update the aforementioned classifier attributes $\vec{w}, \epsilon$ and F. Therefore, the usual delta rule comes into operation. Since this technique is of minor relevance for the remainder of this paper and it is the standard reinforcement strategy of XCS, we refer the reader to [3], [6], [8] for more details.

Initially, $[P]$ is empty. Accordingly, at the beginning of the system's runtime, situations occur where also $[M]$ is empty, i.e. no classifier has yet been created to cover the current situation vector $\vec{x_t}$. For such situations, the *covering* mechanism creates a matching classifier with a suitable condition C and predefined initial values for the other attributes. In certain time intervals a steady-state (GA) is applied on $[M]$. Due to the application on $[M]$, a niching effect occurs that, in combination with deleting classifiers from $[P]$ whenever $|[P]|$ exceeds the maximum size N, exerts generalization pressure [3]. In the remainder of this paper, we will propose strategies to

incorporate interpolation into the algorithmic steps of covering and the GA. Thus, we dive into more detail in Section III-C and III-D.

B. Major Extensions to the Initial XCSF

As already mentioned, we restrict our elaborations to the rudimentary form of XCSF [6]. Since its first proposal in 2001, plenty of research was concerned with improving the performance and capabilities of XCSF. Butz in [7] presents a novel approach for partitioning the problem space. Instead of the conventional hyper-rectangular representation as already described in this paper, he first modified the geometric interpretation toward hyper-spherical shapes, since especially in smooth functions with a certain degree of curvature, the corners of hyper-rectangles cause inaccurate predictions. Based on that, he introduced hyper-ellipsoidal conditions where a distinction in stretching for each dimension is possible. To face the issue of oblique surfaces in the function space, Butz further introduced an approach to allow explicit rotations of the evolved hyper-ellipsoids. Later on, Butz et al. generalized the former approach by omitting the necessary transformation matrix and replace it with an angular representation to avoid disruptive effects during mutation [9]. In this paper, the authors also combined their approach with a novel technique for the reinforcement of the linear approximation. They incorporated the *recursive least squares* method, initially applied to XCSF in [10], to replace the usual modified delta rule for the update of the weight vector $\vec{w}$. In [8], the same authors introduced a methodology to lessen the size of the final population. With the presented *compaction* approach, XCS is able to decrease its population size by 90% on average at the expense of only a short and marginal decrease of the system performance. Another extension introduced by Butz et al. in [11] faces the problem of detrimental forgetting effects when the deletion mechanism is applied on the entire population as usual. Especially in problems where a non-uniformly distributed sampling of the underlying function appears, important but rarely sampled niches can be 'forgotten' due to a higher deletion probability. The *local deletion* mechanism lessens this effect for functions with specific characteristics. Lanzi et al. investigated higher-order polynomials for local approximation in XCSF in [10]. They showed that polynomials beyond the 1st order can improve both the accuracy and the generalization capabilities, and may result in a more compact solution in contrast to standard XCSF with linear approximation. In their work reported in [12], Stalph and Butz introduced the concept *guided evolution* to XCSF. A so-called guided mutation operator is presented that aims at guiding the mutation of classifier conditions toward most suitable condition structures. Therefore, each classifier is extended to hold a set of past samples it has matched. On the basis of an accuracy-weighted covariance matrix, the guided mutation operator is able to identify dimensions with lower, or rather higher impact and in consequence to guide the evolution of the condition structure accordingly. Presented results show that guided evolution leads to an increased learning speed. Additionally, this technique enables XCSF to approximate higher-dimensional functions.

In consequence of the capital achievements of Butz and Lanzi et al. during the last decade, today a fresh impetus is given to XCS research, and there are still efforts in extending and improving its general capabilities. For a broader insight of current XCS research, we refer the interested reader to a recent special issue [13] by virtue of the 20th anniversary of XCS. For a more detailed description of XCS and all relevant extensions, we refer the reader to [3], [5], [6], [8].

III. Local Interpolation

Interpolation methods can be divided in two categories [14]: *Global* and *local* methods. Global methods consider all available sampling points. In contrast, local methods interpolate on the basis of a subset of all samples. We deem local interpolation techniques especially valuable for incorporation into XCS as they reduce the additional computational effort in comparison to global methods. Furthermore, we focus on techniques that weight sampling points that are closer to the unknown situation, or query point $\vec{x}_q$, stronger than samples with a higher distance within the situation space X.

The following paragraphs elaborate on the idea of using interpolation within XCS' algorithmic structure. Therefore, we first describe a particular representative from the variety of local interpolation techniques — a certain modification of the original '*Shepard's Method*' [15]. Afterward, we discuss the fundamental thoughts and interpretative parallels between classifiers and sampling points that allow for integrating interpolation into XCSF. Eventually, we present two approaches that extend XCSF's covering mechanism as well as the GA.

A. Modified Shepard's Method

The name *Modified Shepard's Method* (MSM) is a generic term that summarizes various extensions of Shepard's original method, also called *Inverse Distance Weighting* (IDW). Shepard defined IDW for bivariate interpolation in [15]. For n dimensions, IDW can be formalized as follows [16]:

$$\tilde{f}(\vec{x}_q) = \frac{\sum_{i=1}^{n} f_i W_i}{\sum_{i=1}^{n} W_i}. \tag{3}$$

$\tilde{f}$ denotes the interpolated function value for the n-dimensional query point $\vec{x}_q = (x_1, x_2, \ldots, x_n)$. f_i represents the function value of the i-th sampling point $s_i = (\vec{x}_i, f_i)$ and W_i is a weighting factor that determines to which degree f_i influences the result of the interpolation. There are several opportunities for the choice of the weight function to determine W_i. It depends on the weight function whether the interpolation appears to be local or global. A trivial weight function for yielding a local IDW is given by (cf. [16]):

$$W_k = \begin{cases} 1 & \text{, if } (r - d_i(\vec{x}_q)) > 0 \\ 0 & \text{, else} \end{cases} \tag{4}$$

where $d_i(\vec{x}_q) = \|\vec{x}_q - \vec{x}_i\|_2$ defines the *Euclidean Distance* ($_2$ indicates the L2-norm) between the query point $\vec{x}_q$ and the i-th sampling point's coordinate vector $\vec{x}_i$, and r denotes the

radius from $\vec{x}_q$. A disadvantage of this weight function is that the actual weight W_i is not dependent on the distance from $\vec{x}_q$. To avoid this issue, the weight function should explicitly incorporate the radius. Equation 5 [17] accomplishes that.

$$W_i = \left[\frac{(R_w - d_i(\vec{x}_q))_+}{R_w \cdot d_i(\vec{x}_q)} \right]^2 \quad (5)$$

$R_w > 0$ defines the radius within which encompassed sampling points are considered for the interpolation. Formally, this is guaranteed by the definition of $f_+ = \max(0, f)$. Accordingly, the interpolation appears to be a local method.

An adequate choice of R_w is crucial for an effective application of MSM. If the radius is chosen too large, the MSM possibly degenerates to a global method. In contrast, if it is chosen too small, maybe too few sampling points are considered for the interpolation.

Franke et al. proposed to choose the radius R_w dependent on the parameter N_w [18]. Thereby, N_w is the desired number of sampling points that should be used for the actual interpolation. This dependency can be formalized as follows:

$$R_w = \frac{D}{2}\sqrt{\frac{N_w}{N}}, \quad (6)$$

where $D = \max_{i,j} d_i(x^{(j)})$ and N is the amount of all available sampling points. Thacker provides a summary of suggestions for the right choice of N_w in [16]. We deem such a dynamical choice of R_w to be a crucial issue, especially for problem spaces where the bounds are not known a priori.

B. Interpolation Component: Integrating XCSF and MSM

In our previous work [19], we presented a novel component to extend XCS' algorithmic structure, namely the *Interpolation Component* (IC). Two different architectural approaches to integrate the IC with XCS are discussed. The first approach, XCS-IC, follows the loose-coupling idea. More precisely, the IC maintains a separate set of sampling points SP which allows for a modified action selection regime, the *Action Selection Integration* (ASI) strategy. ASI considers interpolated 'knowledge' from the IC to decide which action to choose from the PA. The main drawback of this approach is that only system-wide values can be interpolated, such as the action $a \in A$ to be executed, or rather the system prediction $P(a)$.

With our second approach, XCS-CIC[1], we go a step further. On the one hand, we allow for interpolation of values on the classifier-level. On the other hand, a further issue, i.e. learning at two places (the population $[P]$ and the set of sampling points SP), is avoided. To achieve this, we interpret the existing classifiers cl_i as sampling points s_i and thus define $SP \subseteq [P]$ where $s_i \in SP$ presents the i-th sampling point for $i = (1 \dots |SP|)$. The remainder of this paper focuses on the XCS-CIC approach, and extends it to be applicable on the XCSF. Accordingly, in the following the IC-extended XCSF is denoted by XCSF-CIC.

[1]CIC results from the leading letters of *Classifer-based Interpolation Component*

Since $cl_i \in [P]$ constitute the sampling points $s_i \in SP$, for simplicity we define $s_i = cl_i$, where a coordinate $\vec{x}_i \in X \subseteq \mathbb{R}^n$ of s_i is represented by the center point of a classifier's hyper-rectangular condition $C = (\vec{l}, \vec{u})$, i.e. $x_j = l_j + (u_j - l_j)/2$ for $j = 1 \dots n$. By using the center point of $cl.C$ as coordinate for s_i, the geometrical condition structure is easily exchangeable, e.g. by hyper-ellipsoids. Since we defined a sampling point s_i to represent a tuple $(\vec{x}_i, f_i)$, the question remains, which attributes serve as function values f_i. By interpreting cl_i as sampling point, any numerical attribute managed by a single classifier cl_i can be used as function value. Due to that and for simplicity, we further define $\vec{f_i} = (w_0 \dots w_n, F, exp)^T$ as a function value vector comprising a classifier's attributes to be interpolated.

With the aforementioned thoughts and definitions at hand, we will now present two strategies for integration of local interpolation into two of XCSF's most essential algorithmic steps — the covering mechanism and the GA.

C. Covering Initials Integration (CII)

As briefly sketched in Section II, covering occurs whenever $[M]$ does not contain any matching classifier for the current situation $\vec{x}_t$. In consequence, XCSF generates a novel classifier cl_{cov} to cover the yet unexplored environmental niche. The condition of the novel classifier $cl_{cov}.C$ is initialized with $\vec{x_t}$ and for each dimension $i = 1 \dots n$ a so-called interval predicate (l_i, u_i) is calculated as follows: $l_i = \max\{l_i^*, x_i - U_{[0,r_0)}\}$ and $u_i = \min\{u_i^*, x_i + U_{[0,r_0)}\}$. $U_{[0,r_0)}$ returns a uniformly distributed random number between 0 and r_0 excluded, where r_0 is a predefined default spread parameter. l_i^* and u_i^* denote the minimal and maximal bounds of the problem space for the i-th dimension, respectively. The remaining attributes are initialized with predefined initial values. $cl_{cov}.\epsilon = \epsilon_I, cl_{cov}.F = F_I$ and the weight vector for the linear approximation $\vec{w}$ is initialized with $w_i = U_{[-1,1]}, i = (1 \dots n)$ [6]. If known, the offset weight w_0 is initialized randomly in between the value range of the function f to be approximated. Otherwise, it is also initialized randomly with $w_0 = U_{[-1,1]}$. Other works on XCSF assume that the actual output value $y = f(\vec{x}_t)$ is immediately known when the situation vector $\vec{x}_t$ arrives, i.e. each trial XCSF receives a complete sample [20]. Accordingly, the offset weight can be set to the actual output value, i.e. $w_0 = f(\vec{x}_t)$, and the initial prediction error drops. This is a plausible assumption, when XCSF is used to approximate a known function. In contrast, our work focuses on the challenge of functions that are unknown during the system's learning phase. Thus, at time t, XCSF is not facing a complete sample $s_t = (\vec{x}_t, f(\vec{x}_t))$, but only the situation vector/sensory input $\vec{x}_t$. We interpret the immediate reward r_{imm} to represent the actual $f(\vec{x}_t)$, as Figure 1 suggests.

The problem with initial values is that they are set arbitrarily to a certain degree. Thus, we deem the incorporation of already existing knowledge valuable to overcome this challenge. A first hint can be acquired from neighboring but non-matching classifiers in the direct proximity. Consider a large problem space with high dimensionality and non-uniform distributed

sampling of the function to be learned. At the beginning, the covering mechanism acts frequently but a complete coverage of the entire problem space is unlikely due to the non-uniform sampling and the high-dimensional space. When a situation arises the first time, the population contains no information about this environmental niche, but possibly about surrounding niches that are sampled more frequently and thus can be expected to predict accurately to a certain degree. With the incorporation of this available knowledge, we expect a more appropriate initialization of newly generated classifiers.

Diving into more detail, the *Covering Initials Integration* (CII) replaces a couple of XCSF's design parameters, e.g. F_I as well as the initialization ranges of the weight vector $\vec{w}$. Instead of assigning $\vec{w}$, F and exp with predefined initial values, the IC is enabled to interpolate these values. Equation 7 exemplarily formalizes the interpolation of a classifier's attributes comprised by the function value vector $\vec{f}_{cov}$:

$$\vec{f}_{cov} = \frac{\sum_{i=1}^{|SP'|} W_i \cdot \vec{f}_i}{\sum_{i=1}^{|SP'|} W_i}, \text{ with } cl_i \in SP' \tag{7}$$

The function value vectors $\vec{f}_i$ of the classifiers cl_i are multiplied component-wise with the corresponding weight W_i. Further, we define $SP' \subseteq SP$, to formalize that not all $cl_i \in [P]$ have to be considered for interpolation. A preselection of experienced and accurate classifiers can be meaningful.

In case of an initially empty population, standard covering serves as fallback solution. As long as there are less than N_w sampling points available in $[P]$, the radius R_w is set large enough to encompass all available classifiers, which equals the global IDW approach.

D. Offspring Initials Integration (OII)

XCSF applies a GA on $[M]$ whenever the mean time since the last GA invocation rises a predefined threshold θ_{GA}. Therefore, each classifier maintains another book-keeping attribute ts that serves as timestamp when the last GA was applied on the match set this particular classifier belonged to. If the GA is activated, two parental classifiers from $[M]$ are chosen by means of *tournament selection* [21]. The parents are copied to form two offspring classifiers cl_{off}. Uniform crossover is applied to the children, i.e. each allele is switched between the two children with equal probability. Thus, crossover alters the geometrical condition structure to explore more suitable forms within the current niche. Mutation modifies the condition probabilistically as described in [6]. In XCS(F)'s standard GA, values for the attributes p, ϵ and F are set to be the mean of both parents, partially discounted by a predefined factor. In contrast, the weight vectors are directly inherited by the parents [6], [10]. Although this is a valid mindset from the genetics perspective, we believe that more than the 'knowledge' of the parents should be involved. This is exactly what the *Offspring Initials Integration* (OII) strategy assures by means of interpolating between the attribute values of all classifiers in the environmental niche (determined by $[M]$). The actual interpolation of the offspring classifiers' attribute vectors $\vec{f}_{off}$ is similar to the CII procedure, as Equation 8 suggests. We note that for the OII, we omitted the interpolation of the weight vectors $\vec{w}_i$, since this, except of certain cases to be investigated more thoroughly, usually resulted in a slightly higher magnitude of $[P]$.

$$\vec{f}_{off} = \frac{\sum_{i=1}^{|[M]|} W_i \cdot \vec{f}_i}{\sum_{i=1}^{|[M]|} W_i}, \text{ with } cl_i \in [M] \tag{8}$$

A major difference between OII and CII is that for OII the set of considered sampling points is defined to be the match set $[M]$, which assures to interpolate within the environmental niche of interest. Standard IDW is also used as fallback solution for OII if $[M]$ contains less than N_w sampling points.

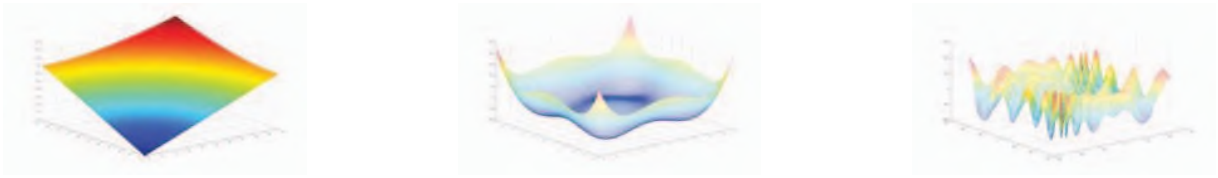
IV. Evaluation

The following paragraphs present the results from the experimental evaluation of the proposed techniques. We conducted experiments on three partially multi-modal test functions with different degrees of dimensionality and curvature. Figure 2 depicts surface plots of the considered functions.

A. Experimental Setup

If not stated differently, we adopted the parameter set[2] suggested by Lanzi et al. in [10]. The *recursive least square* (RLS) method [10], [8] to update the prediction weights was used. Furthermore, *tournament selection* as introduced by Butz et al. in [21] was chosen for parental selection within the GA. The number of considered sampling points was set to $N_w = 19$. All reported results are averages over 30 runs with different random seeds. Each repetition consists of 200000 learning trials. At each trial, XCSF was presented an input vector $\vec{x}(t)$ uniformly sampled from the domains of the test functions. To test our results on statistical significance, we conducted paired t-tests (see Table I). For this reason, the results of all conducted 30 runs were positively tested to follow a normal distribution. The plotted results compare the normalized system error (i.e. an absolute prediction error) and the number of macro-classifiers of standard XCSF and XCSF-CIC. Both metrics are measured and averaged every 100 trials. On the left side of the plots, a log-scale between 0 and 1 is used to better illustrate the interesting ranges of the system error. The number of macro-classifiers is also depicted on a log-scale on the right side of each plot. We adopted this representation from Stalph and Butz (cf. e.g. [12], [24]). Since covering – and thus CII – mainly act at the beginning of a learning task, we restricted our plots to the window of interest for each function to illustrate the potential benefits. However, for the results reported in Table I as well as for the conducted t-tests, we considered the entire sequence of gathered data.

[2] $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 50$, $\epsilon_0 = 0.01$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\epsilon_I = 0.0$, $F_I = 0.01$, $r_0 = 0.1$, $m_0 = 0.2$, $fitnessreduction = 0.1$, $\delta_{rls} = 1000$, $\lambda = 1$



(a) f_1: RMS-like function [6] (b) f_2: Styblinski-Tang function [22], [23] (c) f_3: Eggholder function [23]

Fig. 2: Surface plots of the considered test functions to be approximated for dimensionality $n = 2$

B. Tests on Simple RMS-like Function f_1

In [6], Wilson proposed a rather simple function to test XCSF on higher-dimensional problems with samples drawn uniformly at random from the function's domain. Figure 2a depicts the surface of this function for $n = 2$ dimensions. Wilson describes this function, defined by Equation 9, as a 'RMS-like' function resulting in a flat but rising hyper-plane.

$$f_1(\vec{x}) = [(x_1^2 + \cdots + x_n^2)/n]^{1/2},\ 0 \leq x_i < 100 \qquad (9)$$

Wilson has shown in his article that XCSF is very capable of learning this function even in higher-dimensions ($n = 6$). Thus, we adopted the parameters settings[3] as reported in [6].

Figure 3a shows the effect of CII that occurs mainly at the beginning of the learning phase. The system error drops significantly faster during the first 10000 trials. The population size develops similar but slightly increases during the initial 2000 steps in comparison to standard XCSF. As Figure 3b suggests, in the succeeding learning phase, the number of macro-classifiers drops constantly when compared to standard XCSF. Furthermore, the standard deviation of the system error turns out to be significantly smaller.

Although the 6-dimensional RMS-like function f_1 seems not to be a challenge for XCSF, the utilization of the proposed interpolation strategies show beneficial effects in terms of a reduced system error which is additionally reached more quickly, with less standard deviation, and with a slightly decreased number of classifiers during the learning phase.

Please note that the value range of the RMS-like function is $[0, 100)$. Thus, the targeted error level of 1% corresponds to an absolute error of 1 and the values reported in Table I are above the results of the other test functions.

C. Tests on Curved Styblinski-Tang Function f_2

The second test function, namely the *Styblinski-Tang* function[4] [22], is a common benchmark function from the domain of global optimization [23]. It is characterized by multimodality and shows a certain degree of curvature. Figure 2b shows a surface plot for $n = 2$ dimensions. For n dimensions, the Styblinksi-Tang function is defined by:

$$f_2(\vec{x}) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2},\ -5 \leq x_i \leq 5 \qquad (10)$$

[3] $\alpha = 0.1$, $\beta = 0.1$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 50$, $\epsilon_0 = 0.01$, $\theta_{del} = 50$, $\theta_{sub} = 200$, $\chi = 0.8$, $\mu = 0.04$, $\epsilon_I = 0.0$, $F_I = 0.01$, $m_0 = 0.2$, $fitnessreduction = 0.1$, $\delta_{rls} = 1000$, $\lambda = 1$

[4] w.l.o.g., we normalized domains of definition and the co-domains to $[0, 1]$.

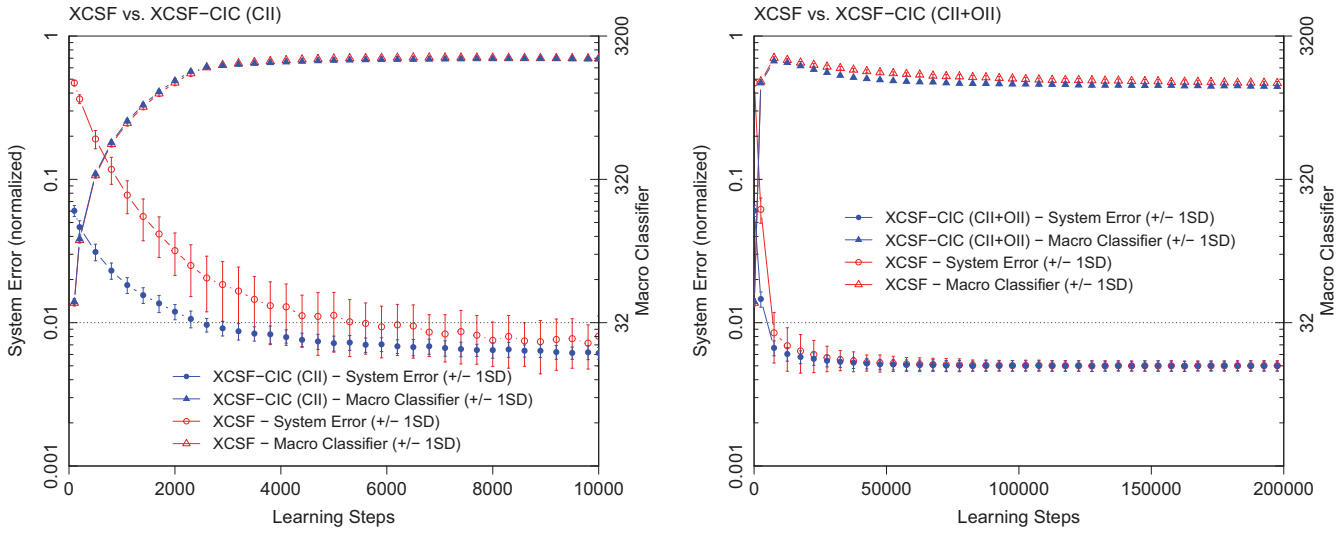
1) 3-dimensional case: We tested the approximation capabilities of XCSF for 3 and 6 dimensions. For the 3-dimensional case, XCSF parameters were set as described by Lanzi et al. in [10] except of $N = 6400$. As for the previous experiment, the interpolation-extended XCSF outperforms standard XCSF in terms of a steeper descent at the beginning of the learning phase (see Figure 4a), a slightly decreased overall system error, as well as an obvious drop of the average magnitude of $[P]$ throughout the entire experiment (see Figure 4b). Overall, the XCSF is not able to reach the target system error of 1%.

2) 6-dimensional case: Parameters were set as in the previous experiment, except of $N = 25600$ and $r_0 = 0.5$ to allow XCSF to explore the problem space effectively and prevent it from falling into a covering-deletion cycle (cf. [20]).

As can be seen in Figure 4b and 5b, XCSF has problems to reach the desired target error level $\epsilon_0 = 0.01$ and to find a suitable approximation of the test function f_2. We attribute this fact to the chosen hyper-rectangular representation for a classifier's condition structure. Butz et al. introduced a more sufficient approach to cope with high curvature in functions [9], i.e. a hyper-ellipsoidal structure. Geometrically speaking, this representation avoids the higher prediction errors that appear in the corners of hyper-rectangles [7]. Furthermore, in [8] the authors enabled the GA to explicitly rotate, squeeze and stretch the ellipsoids within the input space. That allows for a more accurate piecewise approximation of functions that lie oblique in the problem space and have a certain degree of curvature. We are aware of this fact and are currently transferring our proposed techniques to the hyper-ellipsoidal condition representation. With the interpolation strategies activated, only marginal – but nonetheless statistically significant – improvements can be achieved with regard to the average system error. However, the number of necessary classifiers to reach this error level again drops notably, except of the initial learning phase (see Figure 5a). We are not able to exactly explain this behavior yet, but our current efforts are concerned with an in-depth investigation of the classifier evolution when interpolation is incorporated. Apparently, the extend of improvement through interpolation depends on the principal ability of XCSF to learn a certain problem.

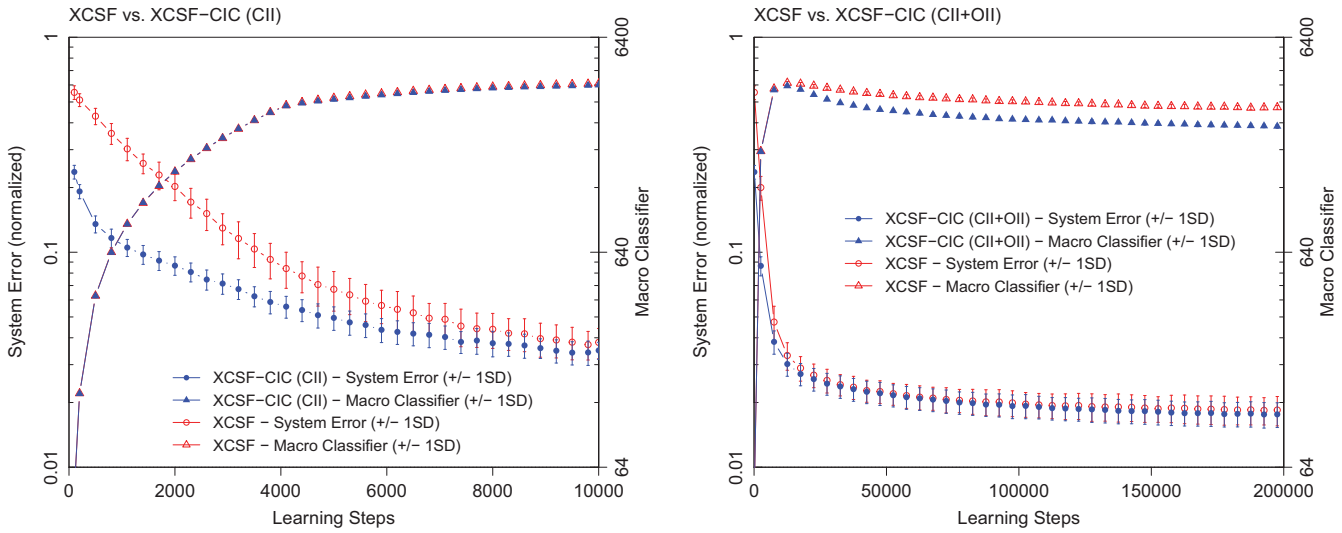
D. Tests on Highly Multi-modal Eggholder Function f_3

The *Eggholder function*[4] [23] serves as another benchmark within the domain of global optimization. It is defined as



(a) (b)

Fig. 3: Results for 6-dimensional RMS-like function f_1



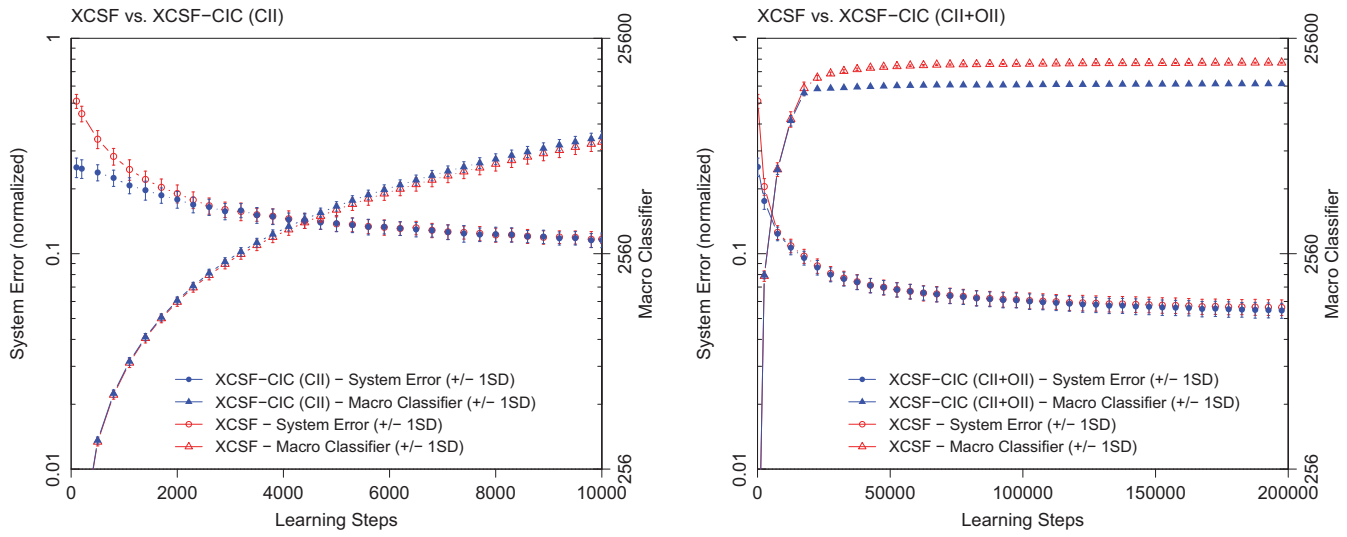
(a) (b)

Fig. 4: Results for 3-dimensional Styblinski-Tang function f_2

follows:

$$f_3(x,y) = -(y+47)\sin\left(\sqrt{\left|\frac{x}{2}+(y+47)\right|}\right) - x\sin(\sqrt{|x-(y+47)|}), \quad -512 \leq x,y \leq 512 \tag{11}$$

Figure 6 illustrates the Eggholder function's characteristics, i.e. a high degree of multi-modality and the severe and repeating curvatures in both dimensions. We configured XCSF as for previous experiments except: $N = 6400, r_0 = 0.05$ and $m_0 = 0.02$. Due to the complex shape of function f_3, XCSF is not able to get a sufficient fitness signal when initially generated classifiers cover a too large area of the problem space. Thus, we followed the recommendation of Butz et al. [8], [24] and chose r_0 and m_0 small enough to guarantee fitness pressure. The conducted parameter study indicated that the rather small chosen values were most suitable. Consequently, we got smaller initial hyper-rectangles to cover the variety of valleys and peaks. Again, the CII strategy accelerates the initial decrease of the system error. In conjunction with OII, the error



(a) (b)

Fig. 5: Results for 6-dimensional Styblinski-Tang function f_2

level drops constantly below the level of standard XCSF. Also the evolved number of macro-classifiers decreases for XCSF-CIC with CII and OII. Again, the target error $\epsilon_0 = 0.01$ was not reached by any of the experiment candidates. As already mentioned above, we assume that this is due to the utilized hyper-rectangular condition representation.

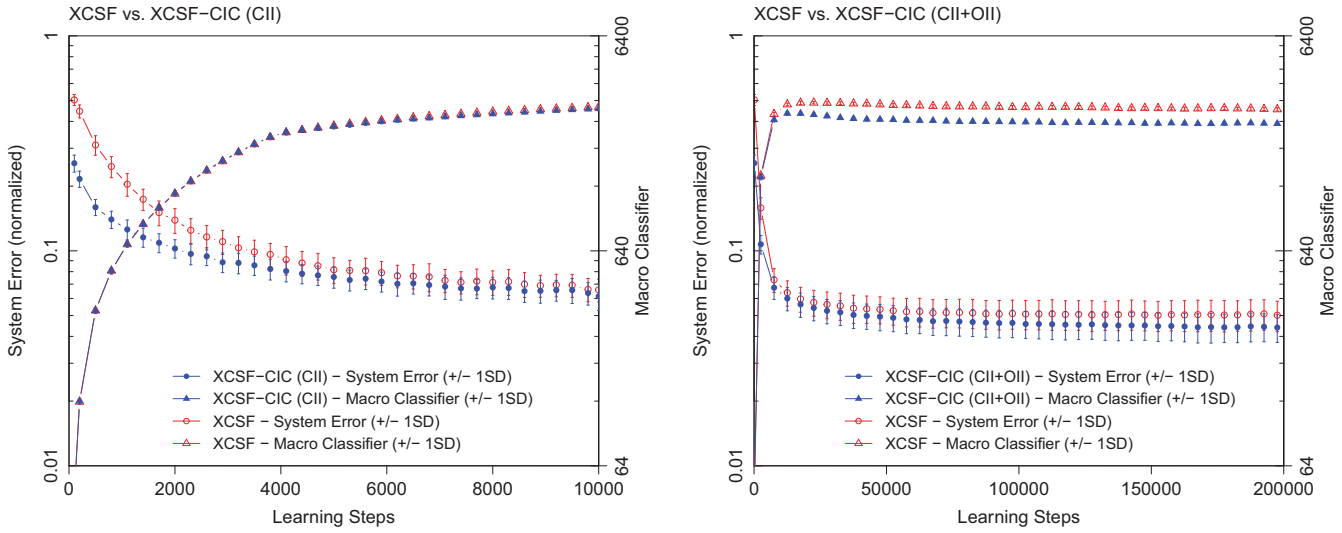
TABLE I: Summary of results on functions f_{1-3}. * (**) indicates statistically (highly) significant reductions of the reported metrics compared to standard XCSF, i.e. that for the p-values of paired one-sided t-tests holds $p < \alpha = 0.05\ (0.01)$.

f_1 **RMS-like 6D** (target error $\epsilon_0 = 1$)	*System Error* mean	*Macro Classifiers* mean
XCSF w/ **CII**	.5394**	1680.62
XCSF w/ **OII**	.6750	1511.06**
XCSF w/ **CII+OII**	.5387**	1547.73**
Standard XCSF	.6690	1676.59
f_2 **Styblinski-Tang 3D**	*System Error*	*Macro Classifiers*
XCSF w/ **CII**	.0227**	3280.61
XCSF w/ **OII**	.0269	2659.01**
XCSF w/ **CII+OII**	.0223**	2766.32**
Standard XCSF	.0260	3250.96
f_2 **Styblinski-Tang 6D**	*System Error*	*Macro Classifiers*
XCSF w/ **CII**	.0689**	18288.88
XCSF w/ **OII**	.0696**	14765.20**
XCSF w/ **CII+OII**	.0683**	14805.32**
Standard XCSF	.0699	18201.20
f_3 **Eggholder 2D**	*System Error*	*Macro Classifiers*
XCSF w/ **CII**	.0520**	2954.93
XCSF w/ **OII**	.0540**	2518.66**
XCSF w/ **CII+OII**	.0491**	2540.76**
Standard XCSF	.0553	2955.01

V. Conclusion

In this paper, we proposed a concept of using interpolation within the algorithmic structure of XCSF to affect the generation of classifiers. Fundamental thoughts about how interpolation can generally be incorporated were discussed. After diving into more detail, we introduced two integration strategies, namely the *Covering Initials Integration* (CII) strategy as well as the *Offspring Initials Integration* (OII) strategy. Based on experiments on three test functions with different degrees of complexity, we showed that the proposed techniques lead to an improvement regarding XCSF's learning speed and its prediction accuracy in terms of a decreased system error and, at the same time, to a smaller average magnitude of $[P]$. The CII method mainly acts at the beginning of a learning task, since covering usually occurs at this time. In general, CII helps the system to faster converge to the desired or rather reachable error level. The OII strategy promises additional performance improvements throughout the entire learning task. However, the extent of achievable improvements by means of interpolation depends strongly on the general ability of XCSF to approximate the considered function, i.e. the better standard XCSF is able to approximate a function, the more value can be expected and observed through interpolation. According to that, future work will investigate the ability of XCSF-CIC to approximate different types of functions as suggested in our previous work [25].

Although we reported promising results in this paper, current research activities focus on a more thorough investigation of the sensibility and (inter-)dependencies of XCSF's standard parameters and the interpolation strategies. Our current efforts are also concerned with the development of further integration strategies, e.g. the interpolation of a classifier's condition



(a) (b)

Fig. 6: Results for 2-dimensional Eggholder function f_3

structure and alignment. According to that, the top priority for us is to transfer our interpolation techniques to rotating hyper-ellipsoidal condition structures as proposed by Butz et al. [8]. Furthermore, we are currently working on an identification and implementation of an XCSF 'state-of-the-art'. The comparison with alternative interpolation techniques constitutes another topic of our research agenda.

REFERENCES

[1] J. H. Holland, "Adaptation," in *Progress in Theoretical Biology*, R. Rosen and F. Snell, Eds. New York: Academic Press, 1976, vol. 4, pp. 263–293.

[2] S. W. Wilson, "ZCS: A Zeroth Level Classifier System." *Evolutionary Computation*, vol. 2, no. 1, pp. 1–18, 1994.

[3] ——, "Classifier Fitness Based on Accuracy," *Evol. Comp.*, vol. 3, no. 2, pp. 149–175, 1995.

[4] T. Kovacs, "XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions," in *Soft Computing in Engineering Design and Manufacturing*. Springer London, 1998, pp. 59–68.

[5] S. W. Wilson, "Get Real! XCS with Continuous-Valued Inputs," in *Learning Classifier Systems*, ser. LNCS. Springer, 2000, vol. 1813, pp. 209–219.

[6] ——, "Classifiers that Approximate Functions," *Natural Comp.*, vol. 1, no. 2-3, pp. 211–234, 2002.

[7] M. V. Butz, "Kernel-based, Ellipsoidal Conditions in the Real-valued XCS Classifier System," *GECCO '05*, pp. 1835–1842, 2005.

[8] M. Butz, P. Lanzi, and S. Wilson, "Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction," *Evol. Comp, IEEE Trans. on*, vol. 12, no. 3, pp. 355–376, 2008.

[9] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Hyper-ellipsoidal Conditions in XCS: Rotation, Linear Approximation, and Solution Structure," in *GECCO*, 2006, pp. 1457–1464.

[10] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg, "Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension," Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign, Urbana, IL, IlliGAL Rep. 2005012, 2005.

[11] M. Butz and O. Sigaud, "XCSF with Local Deletion: Preventing Detrimental Forgetting," *Evolutionary Intelligence*, vol. 5, no. 2, pp. 117–127, 2012.

[12] P. O. Stalph and M. V. Butz, "Guided Evolution in XCSF," in *Proc. of GECCO '12*. New York, NY, USA: ACM, 2012, pp. 911–918.

[13] T. Kovacs, M. Iqbal, K. Shafi, and R. Urbanowicz, "Special Issue on the 20th Anniversary of XCS," *Evolutionary Intelligence*, vol. 8, no. 2-3, pp. 51–53, 2015.

[14] R. Franke, "A Critical Comparison of some Methods for Interpolation of Scattered Data," DTIC Document, Tech. Rep., 1979.

[15] D. Shepard, "A Two-dimensional Interpolation Function for Irregularly-spaced Data," in *Proc. of 23rd ACM National Conference*, ser. ACM '68. New York, NY, USA: ACM, 1968, pp. 517–524.

[16] W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry, "Algorithm XXX: SHEPPACK: Modified Shepard Algorithm for Interpolation of Scattered Multivariate Data," Department of Computer Science, Virginia Polytechnic Institute & State University, Departmental Technical Report TR 09-13, 2009.

[17] R. J. Renka, "Multivariate Interpolation of Large Sets of Scattered Data," *ACM Transactions on Mathematical Software (TOMS)*, vol. 14, no. 2, pp. 139–148, 1988.

[18] R. Franke and G. Nielson, "Smooth Interpolation of Large Sets of Scattered Data," *International journal for numerical methods in engineering*, vol. 15, no. 11, pp. 1691–1704, 1980.

[19] A. Stein, D. Rauh, S. Tomforde, and J. Hähner, "Augmenting the Algorithmic Structure of XCS by Means of Interpolation," in *Proc. of 29th International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: Springer, April 2016, pp. 348–360.

[20] P. O. Stalph, X. Llor, D. E. Goldberg, and M. V. Butz, "Resource Management and Scalability of the XCSF Learning Classifier System," *Theoretical Computer Science*, vol. 425, pp. 126 – 141, 2012, theoretical Foundations of Evolutionary Computation.

[21] M. V. Butz, K. Sastry, and D. E. Goldberg, "Tournament Selection: Stable Fitness Pressure in XCS," in *GECCO 2003*. Springer Berlin Heidelberg, 2003, pp. 1857–1869.

[22] M. A. Styblinski and T. S. Tang, "Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing," *Neural Netw.*, vol. 3, no. 4, pp. 467–483, Jul. 1990.

[23] M. Jamil and X. Yang, "A Literature Survey of Benchmark Functions for Global Optimisation Problems," *IJMNO*, vol. 4, no. 2, 2013.

[24] P. Stalph and M. Butz, "Current XCSF Capabilities and Challenges," in *Learning Classifier Systems*, ser. LNCS. Springer Berlin Heidelberg, 2010, vol. 6471, pp. 57–69.

[25] S. Tomforde, A. Brameshuber, J. Hähner, and C. Müller-Schloer, "Restricted On-line Learning in Real-world Systems," in *Evolutionary Computation (CEC), IEEE Congress on*, June 2011, pp. 1628–1635.